Tangible software modeling with multi-touch technology

Müller, Sebastian ; Würsch, Michael ; Schöni, Pascal ; Ghezzi, Giacomo ; Giger, Emanuel ; Gall, Harald C

Abstract: This paper describes a design study that explores how multi-touch devices can provide support for developers when carrying out modeling tasks in software development. We investigate how well a multi-touch augmented approach performs compared to a traditional approach and if this new approach can be integrated into existing software engineering processes. For that, we have implemented a fully functional prototype, which is concerned with agreeing on a good objectoriented design through the course of a Class Responsibility Collaboration (CRC) modeling session. We describe how multitouch technology helps with integrating CRC cards with larger design methodologies, without loosing their unique physical interaction aspect. We observed high-potential in augmenting such informal sessions in software engineering with novel user interfaces, such as those provided by multi-touch devices.

DOI: https://doi.org/10.1109/CHASE.2012.6223001

Tangible Software Modeling with Multi-touch Technology

Sebastian Müller, Michael Würsch, Pascal Schöni, Giacomo Ghezzi, Emanuel Giger, and Harald C. Gall
s.e.a.l. - software architecture and evolution lab
Department of Informatics
University of Zurich, Switzerland
{*smueller, wuersch, schoeni, ghezzi, giger, gall*}*@ifi.uzh.ch*

***Abstract*—This paper describes a design study that explores how multi-touch devices can provide support for developers when carrying out modeling tasks in software development. We investigate how well a multi-touch augmented approach performs compared to a traditional approach and if this new approach can be integrated into existing software engineering processes. For that, we have implemented a fully-functional prototype, which is concerned with agreeing on a good object-oriented design through the course of a *Class Responsibility Collaboration* (CRC) modeling session. We describe how multi-touch technology helps with integrating CRC cards with larger design methodologies, without loosing their unique physical interaction aspect. We observed high-potential in augmenting such informal sessions in software engineering with novel user interfaces, such as those provided by multi-touch devices.**

***Keywords*-CRC analysis; modeling; multi-touch; collaboration**

I. Introduction

The success of large-scale software development projects is determined by their team members' ability to cooperate. Software engineers have therefore developed a wide range of processes and tools, and adopted many different communication technologies to support coordination between physically separated collaborators. Development is centered around artifacts, such as requirements documents, source code, and problem reports. These artifacts are either stored in version control systems or full-blown software configuration management tools. Such infrastructures succeed in enabling team-work across regional and organizational boundaries.

Less work has been done on supporting and unifying those activities during a software life-cycle that happen face-to-face. In many companies, important decisions are made in meetings, agreements are reached during informal conversations, and knowledge transfer between developers happens during coffee breaks or lunchtime. These conversations can only be augmented very carefully by technological means. Otherwise they run risk of loosing their effortless character which is key to creativity and enables innovation.

Tool support of some kind, however, is still desirable: The outcome of such informal interaction is hardly ever made explicit by writing it down and sharing the resulting documents between all potentially affected stakeholders. Even if the tedious process of formalization is undertaken, people tend to forget important details or introduce errors and ambiguities, considering the plethora of details that go hand in hand with highly complex, large-scale software development.

In this paper, we describe how we use the *Microsoft Surface* multi-touch table as a vehicle to augment a typical modeling task in software engineering. This task is part of an early design phase and concerned with agreeing on a good object-oriented design. Typically, the task is inherently cooperative and it is appropriate to carry it out by a team of two to four domain experts and software engineers who are physically co-located to increase efficiency in communication. While it is hard to augment such a setting with conventional technologies, multi-touch devices provide a benefiting form-factor, as we will outline throughout the remainder of the paper.

II. Tangible Software Modeling: A Design Study

In the following, we describe a design study that we have carried out to explore the usefulness of touch-table devices for a common software modeling task. For that, we have implemented a fully-functional prototype and have deployed it onto a *Microsoft Surface* device.

In our design study, we aim to support software engineers in an early design phase, *i.e.* when they are concerned with agreeing on a good object-oriented design. This usually happens after the initial requirements have been collected and are documented, often by means of text documents.

A. Background

Class Responsibility Collaboration (CRC) cards have originally been introduced by Beck and Cunningham in [1] to teach both novice and experienced programmers how to articulate a comprehensive model of an object-oriented system. The method recently has re-gained popularity, *e.g.* within the agile development movement. There are several variants and extensions but in its simplest form, the method only relies on physical *index cards*.

One participant reads out loud a requirements specification, for example, a use-case description. Nouns appearing in the text are candidates for *Classes* and are written onto an index card as the class name. Newly created classes, *i.e.* index cards, are assigned to a participant, who is then in charge for it for the remainder of the session. Verbs in the

text are candidates for being a *Responsibility* of the noun or class to which they belong to. Participants write down the responsibilities on the left side of their according index card. Whenever a class can not fulfill its responsibilities on its own, it has to establish a *Collaboration* with another class. These are noted on the right side of the index card(s). Cards can then be physically re-arranged, *e.g.* to denote some kind of relation with physical proximity.

B. Intent

Beck and Cunningham especially acknowledged the value of physical interaction for the sake of improving the object-oriented understanding, which had led them to oppose a computerization of the cards. The downside they mentioned was that integrating the cards with larger design methodologies and with particular language environments, without loosing the unique physical interaction aspect, would only be possible with a new kind of user interface, far beyond the state of the art at that time [1].

Two decades later, devices, such as the *Microsoft Surface* touch-table, are starting to see a wider distribution. Multi-touch user interfaces are capable of presenting digital artifacts in a way that users can intuitively interact with, similarly as they would with physical objects.

C. Principles for Tangible Software Modeling

To provide developers with an approach that can outperform a traditional paper-based approach, our prototype implementation relies on three principles: a tangible user interface, multi-touch augmented interaction and process support. The user interface basically consists of digital index cards and physical, tangible cubes that can be used to interact with the application. Process support includes automatic processing of requirements documents before the CRC modeling session as well as the incorporation of our approach into further software development steps.

D. Design Support

Our prototype implementation stays as close to the original CRC idea as possible while carefully digitalizing the process and its artifacts. A screenshot of the prototype running in the simulator of the *Surface SDK* is given in Figure 1.

Previous to the CRC analysis session, participants can send requirements documents, *e.g.* use-case descriptions in the PDF or MS Word format to the touch-table. This can be done using email: our prototype regularly polls for messages sent to its dedicated email account and downloads the attachments automatically.[1] The documents are then scanned by WordNet [2] to remove stop words. Only nouns and verbs are kept and classified accordingly. Subsequently, words are

[1]Future versions of the MS Surface might even allow to place sheets of paper directly on the screen for scanning. The resolution of the MS Surface 1.0, however, does not suffice for optical character recognition.

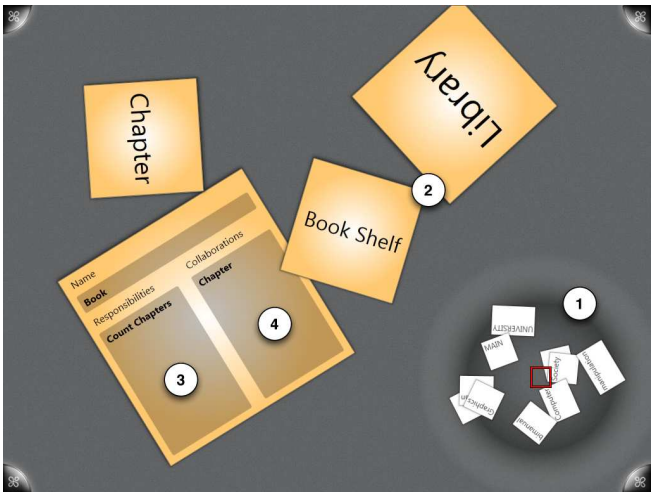

Figure 1. Class Responsibility Collaboration analysis. The numbers denote different parts of the user interface and are described in the text in more detail.

weighted according to how often they occur in a document – the higher the number of occurrences, the higher their weight. The premise is that words mentioned more often in the requirements documents are likely to be important in the domain. These weightings are used to generate tag clouds from which the participants can conveniently choose words to name cards and responsibilities.

A CRC analysis session starts with a blank screen. Participants can place a brown[2] wooden cube anywhere on the touch-table. The cube has a barcode-like pattern, a so-called *Surface tag*, at its bottom. The barcode is recognized by the device and triggers the display of a tag cloud around the location where the cube has been placed (Figure 1, denoted by 1). The cloud contains the most prominent nouns extracted from requirements documents. Dragging one of the nouns away from the immediate perimeter of the cloud automatically creates and displays a new card, *i.e.* a class (Figure 1, denoted by 2). The noun is used for the class name. Similarly, an orange cube brings up a tag cloud with verbs. Participants can then drag a verb onto a card to assign a responsibility to a class. Touching two cards simultaneously for a couple of seconds (without moving them around) establishes a collaboration between them.

Cards can be freely moved around and rotated on the screen. By default, only the card name is shown and the card is resized automatically to take up as little space as possible on the screen. Participants, however, can use a pinch-to-zoom gesture to enlarge cards. When a card reaches an adequate size, two columns become visible: the left one lists

[2]In Switzerland, children in school often have to mark different parts of speech with different colors when they are taught their first grammar lessons. Nouns usually are marked with a brown color, verbs are denoted in orange.

all the responsibilities (Figure 1, denoted by 3), whilst the right one lists the card's collaborators (Figure 1, denoted by 4). It is then also possible to manually enter responsibilities and collaborators, respectively. For this, a virtual keyboard is brought up, which is useful whenever, *e.g.* a verb is missing in the tag cloud. Optionally, new cards can be created also this way.

There are several additional features that are not key to the CRC analysis but come in handy, for example if scalability is needed: Cards can be stacked to group them into *packages* by dragging one or several cards onto another one. It is also possible to tag cards and hide them subsequently if they are not used at the moment. In addition, all collaborators can be gathered around the currently selected card. A black wooden cylinder – the so called black hole – can be placed on the touch-table and acts as a trash bin. Dragging cards onto the black hole disposes them.

When the CRC analysis session is completed, the prototype generates C# code from the outcome of the session, creates a zip-archive, and distributes it via email to the participants for further use. Each card will trigger the creation of a class. Packaging information is also considered, if available. Responsibilities are translated into parameterless methods with `void` as their return-types. For each collaborator, an attribute of the collaborator's type is added to the class, in addition to a pair of suitable accessor-methods. To sustain traceability from requirements to code, all generated source code entities are automatically annotated with the file names of the documents they originate from (if applicable). This is possible because the prototype keeps track of the origin of nouns and verbs whenever they are chosen from the tag clouds as class and responsibility names, respectively.

Based on our observations during a preliminary study and our own experience, we emanate that conducting the CRC analysis on a touch-table bears several benefits – while preserving the unique features of the physical index cards that have been promoted by Beck and Cunningham. Assuming that the participants have already a rough initial understanding of the problem domain, the analysis will quickly gain momentum: The natural language preprocessing done on the requirements documents relieves the participants to a certain extent from going through all the requirements documents again during the session. The automatic generation of code stubs provides developers valuable support to convert the information on the physical index cards into a machine-readable state. Further, the automatic preservation of links between the concepts in the code and the requirements documents promotes requirements traceability.

On the other hand, the approach quickly reaches its limits, when no requirements documents are available. While it is possible to manually enter identifiers with the help of a virtual keyboard, the need of entering text manually should be limited to a minimum. Otherwise, according to our experience, usability clearly suffers, especially when compared to a conventional keyboard and mouse experience. Other input methods, *e.g.* voice recognition, suggest themselves but are hard to incorporate in a way so that they do not disturb the natural interaction between collaborators, *e.g.* when a fruitful discussion has to be interrupted just to provide speech input. We are experimenting with different approaches to overcome these issues; one is to record every spoken word during a session and to translate it to text immediately, potentially filtering stop words on-the-fly. The words spoken during the last 20 to 30 seconds can then be displayed on screen at any given point in time, or the participants can move forward and backward through the transcript using a sliding window approach, selecting words for card names, responsibilities, etc., when appropriate.

III. Preliminary Evaluation

We performed a preliminary evaluation of our prototype implementation to assess the capabilities of our approach and to explore how different aspects of our approach are perceived. In the future, we plan to conduct an observational study in combination with interviews to gain further insights on how our approach is used in detail.

A. Study Settings

We investigated the following research questions:

1) How well is the multi-touch augmented approach received compared to traditional, paper-based index cards?
2) Can our prototype be integrated into existing software engineering processes without difficulties?
3) Are there features of our multi-touch augmented approach that are particularly liked or disliked by its users?

To answer these questions, we have conducted a user study with twelve participants. The participants had an academic background, including undergraduate and graduate students, as well as senior research associates, with some having also extensive industrial experience.

The subjects were split into teams of two – to also incorporate the aspect of collaboration. Each team had to solve two exercises: In Exercise I, a short textual description of a library design was given. Similarly, Exercise II was based on an equally short description of a hospital information system. Each exercise was either to be solved with traditional index cards or with our prototype.

We used a counterbalanced design for our user study to compensate for learning effects from exercise to exercise, as well as for subject abilities. This means that we varied both – whether a team had to solve first Exercise I then II, or vice versa, and whether they first had to use traditional index cards and then our multi-touch augmented approach, or vice versa.

Following the experiment, the participants had to fill out a questionnaire. The questionnaire asked the participants

to rate their favourite approach for different aspects of a CRC modeling session. The rating was based on a five-point Likert scale, with one being "multi-touch approach" and five being "traditional index cards". Additionally, the participants were asked to rate basic statements about our approach on a five-point Likert scale with one being "strongly disagree" and five being "strongly agree". Finally, in an open questions part, the participants could write down the aspects of our approach they liked most, didn't like and suggestions to improve it. We neither assessed the quality of the solution, nor the time it took the teams to come up with it.

B. Results

The results of this evaluation are shown in Tables I and II. When comparing our multi-touch implementation with traditional index cards, the participants reported that both approaches are equally easy to use. However, most participants agreed that the big advantage of our multi-touch approach is the possibility to easily rearrange cards, using natural finger movements. Not surprisingly, the digitalization of the index cards has convinced the participants that the outcome can be used in further development steps, without the need to digitalize hand-written cards. As a consequence, most participants would prefer our multi-touch approach to traditional index cards. Table II shows that most participants like the interaction principles used by our prototype. They agree that *Surface tags* could outperform traditional UI menus.

One of the most important advancement of our application is the incorporation into the software engineering process. The results for Statement 4 in Table II suggest that the participants think that it is very useful to pre-process requirements documents as input for the CRC modeling session. Likewise, the answers to the open questions part show that it was mentioned frequently that the automatic generation of code and the requirements traceability is considered to be a major advantage over traditional index cards by the participants of our study.

Table I
COMPARISON OF MULTI-TOUCH AUGMENTED CRC ANALYSIS AND TRADITIONAL INDEX CARDS ON A FIVE POINT LIKERT SCALE (1 = MULTI-TOUCH / 5 = INDEX CARDS)

Question	Mean
For what approach, the first steps to start an analysis session were more clear?	2.83
Which approach is easier to use?	3.00
Which of the two approaches supports collaboration better?	2.58
Which approach provides more flexibility, e.g. when arranging cards?	2.58
Looking at the outcome of the analysis, which approach makes it easier to re-use the results for further development?	1.75
All in all, which approach do you prefer?	2.45

Table II
HOW THE PARTICIPANTS RECEIVED THE MULTI-TOUCH AUGMENTED CRC ANALYSIS APPROACH ON A FIVE POINT LIKERT SCALE (1 = STRONGLY DISAGREE / 5 = STRONGLY AGREE)

Statement	Mean
The application is easy to use.	3.42
I prefer using wooden cubes over a traditional menu paradigm.	4.18
Setting up a collaboration between two cards is straightforward.	4.08
The process for providing PDF documents as input is convenient.	4.67
The tags proposed by the application influenced my design.	3.42

IV. RELATED WORK

Related work with respect to our approach can be broadly divided into two categories: approaches that generally foster collaborative software engineering and approaches using multi-touch enabled devices to support typical software engineering activities.

A comprehensive overview of the state of the art in collaborative software engineering is given in [3]. The author reasons about the potential of new technologies and user interface paradigms in general to enhance collaboration tools, without specifically considering multi-touch devices.

Multi-touch enabled approaches are not yet widely used to support typical software engineering activities. Most approaches support some form of agile planning meetings, provide awareness or support typical software visualization tasks.

Frisch *et al.* have studied how users prefer to work with graphical modeling tools on an interactive tabletop in [4]. Their focus was on identifying what hand and pen gestures come natural to users when they sketch and edit diagrams. Their approach does not attempt to produce, or use, any meaningful artifacts in the context of software engineering.

In [5] an application for tabletops is presented that can be used in agile project planning meetings. Both co-located and distributed teams are supported. The application provides means to create, move and pile index cards. But it is not possible to automatically create these cards or incorporate them into further development processes.

Ghanam *et al.* [6] describe a planning tool that should support synchronous planning meetings without breaking the natural interaction of the team. For that purpose the tool provides an orientation-independent user interface that can be used to create, move and modify story cards. Other actions are not supported.

In [7], a tabletop-based project planning tool is introduced. The feature set is limited to creating *story cards* – widely used in agile development – passing or tossing the cards from one collaborator to another, resizing, or rotating them. Neither does the prototype provide automated support for creating these cards based on, *e.g.* requirements docu-

ments, nor does it integrate with further software engineering process steps.

Hardy *et al.* [8] present an approach that supports awareness by using an interactive desk, called *CoffeeTable*, that can be used as a shared workspace by multiple software developers. Hardy *et al.* argue that using this shared workspace can foster the collaboration between its users when working synchronously on the same software project.

Finally, Anslow *et al.* [9] present an approach to support software visualizations for code exploration in a co-located and collaborative environment using multi-touch technology. Their approach applies polymetric views that can be manipulated using finger gestures to visualize and explore source code entities of a software system.

V. Conclusion

In this paper we have presented a multi-touch augmented approach that supports developers when carrying out CRC modeling tasks. We have investigated how well our approach is received by its users compared to traditional, pen and paper-based index cards. A first evaluation showed that the prototype implementation is well-accepted by software engineers and that our approach can be incorporated into existing software engineering processes without difficulties. Concerning the user experience, it became obvious that our prototype implementation provides its users with a big enough feature set to solve typical CRC modeling tasks.

While multi-touch interfaces recently have gained tremendous momentum in the consumer market, the technology has not yet received much attention in the context of software engineering – neither from a practitioner's, nor from a researcher's point of view. The reasons for this dichotomy are manifold: Product marketing is targeted towards consumers, typically people with non-technical background that are more easily impressed by playful interfaces, whereas software engineers often care more about information richness and utility than presentation. Touch interfaces, while commonly considered more intuitive, are often less precise than the combination of mouse and keyboard. Experienced users of personal computers feel less productive and miss the tactile feedback of key-presses and mouse clicks. Especially entering longer text is cumbersome.

The utility of multi-touch devices is probably less a question of raw technology. More important is that the *form follows function* – devices, such as the *Microsoft Surface*, clearly have a form-factor that, in contrast to ordinary computer monitors, comes natural when people meet face-to-face to solve problems in the software engineering domain collaboratively. There is high-potential in augmenting these informal sessions with adequate information technology – one of the most promising perspectives is that digital artifacts gain a unique physical interaction aspect while preserving machine readability. In consequence the need of cross-media conversion, *e.g.* from hand-written meeting minutes to source code, is likely to decrease.

References

[1] K. Beck and W. Cunningham, "A laboratory for teaching object oriented thinking," *SIGPLAN Notices*, vol. 24, pp. 1–6, 1989.

[2] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, pp. 39–41, 1995.

[3] J. Whitehead, "Collaboration in software engineering: A roadmap," in *2007 Future of Software Engineering*, FOSE '07, (Washington, DC, USA), pp. 214–225, IEEE Computer Society, 2007.

[4] M. Frisch, J. Heydekorn, and R. Dachselt, "Investigating multi-touch and pen gestures for diagram editing on interactive surfaces," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, (New York, NY, USA), pp. 149–156, ACM, 2009.

[5] R. Morgan and F. Maurer, "Maseplanner: A card-based distributed planning tool for agile teams," in *Proceedings of the IEEE International Conference on Global Software Engineering*, (Washington, DC, USA), pp. 132–138, IEEE Computer Society, 2006.

[6] Y. Ghanam, X. Wang, and F. Maurer, "Utilizing digital tabletops in collocated agile planning meetings," in *Proceedings of the Agile 2008*, (Washington, DC, USA), pp. 51–62, IEEE Computer Society, 2008.

[7] X. Wang and F. Maurer, "Tabletop agileplanner: A tabletop-based project planning tool for agile software development teams," in *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pp. 121–128, 2008.

[8] J. Hardy, C. Bull, G. Kotonya, and J. Whittle, "Digitally annexing desk space for software development (nier track)," in *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, (New York, NY, USA), pp. 812–815, ACM, 2011.

[9] C. Anslow, S. Marshall, J. Noble, and R. Biddle, "Co-located collaborative software visualization," in *Human Aspects of Software Engineering*, HAoSE '10, (New York, NY, USA), pp. 4:1–4:2, ACM, 2010.